Numerical Analysis (3/7)
integration & differentiation

University of Luxembourg - 2024

Master in Mathematics

Hadrien Beriot

Outline

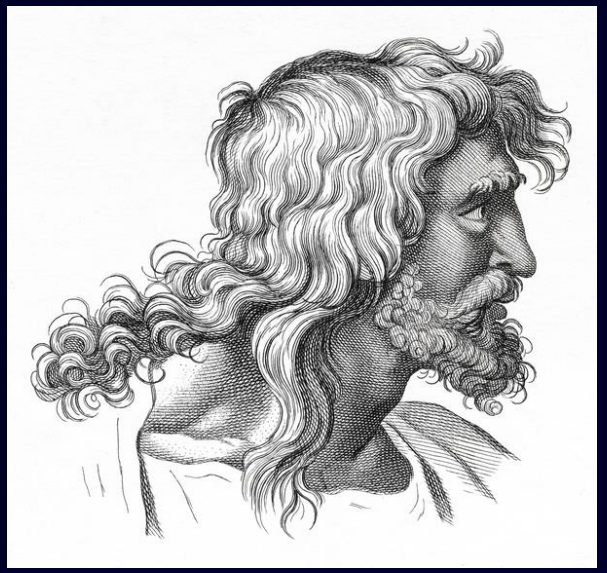
1. Numerical integration
2. Numerical differentiation
3. Summary

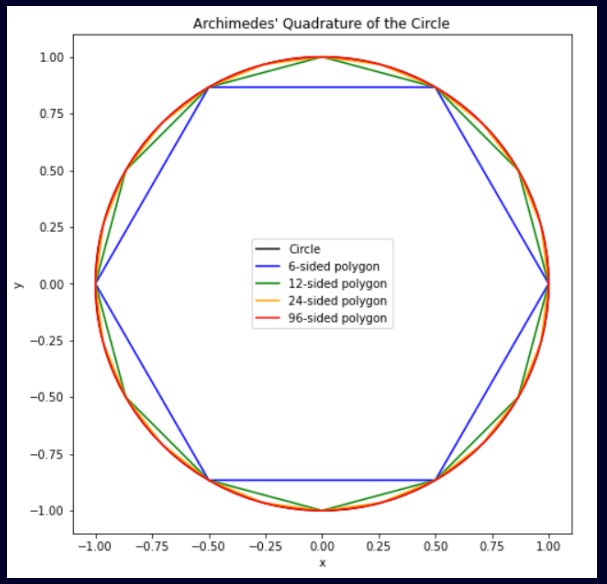
1. Numerical Integration

Dates back from Antiquity

- In antiquity, aera approximation was know.
- Approach was to tile the region with small squares and then count the number of squares within the region
- Archimedes (287–212 BCE) used **polygons** to approximate the area of a circle.
- By calculating the areas of inscribed and circumscribed polygons, he found an early approximation for π
- He obtained by using 96 polygons

$$3.1408 < \pi < 3.1429$$





Numerical integration

- Being given a continuous function $f : [a, b] \to \mathbb{R}$, we try to compute the integral

$$I = \int_a^b f(x)\, dx$$

- We need to use **numerical integration** when we do not know the explicit form of the primitives of f :

$$f(x) = e^{-x^2}, f(x) = \frac{e^x}{x}, f(x) = \frac{1}{\log(x)}$$

- We do not want to compute exactly the value of this integral but only an approximate value I_{app} with an a priori given accuracy ϵ i.e. such that

$$|I - I_{app}| < \epsilon$$

Basic principle

- Once again, we “replace” the function f by a close enough function for which computing its primitive is quite easy → **polynomial interpolation**
- However, in some occasions we must be careful with global polynomial interpolation
- This is why we will most of the time adopt a **piecewise polynomial approximation**

Description of the principle

- We subdivide the interval $[a, b]$ in some smaller intervals $a = x_0 < x_1 < \cdots < x_N = b$
- By linearity we have

$$\int_a^b f(x)dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx$$

- Hence, we are led to compute some integrals for which the length of the integration interval is relatively small.

Change of variable

- For each integral, the following change of variable

$$x = \frac{(x_{i+1} - x_i)s + (x_i + x_{i+1})}{2} = \frac{h_i s + (x_i + x_{i+1})}{2} \quad \text{with} \quad h_i = x_{i+1} - x_i$$

- yields

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h_i}{2} \int_{-1}^{1} g_i(s)ds$$

- setting

$$g_i(s) = f\left(\frac{h_i s + (x_i + x_{i+1})}{2}\right) \quad \forall s \in [-1, 1]$$

- Therefore, we focus on the approximation of integrals on the **reference interval** $[-1, 1]$

Interpolating the integrand over [-1,1]

- Being given n points $(s_j)_{1\leq j\leq n}$, $n > 0$ in $[-1,1]$, we approximate the function g by the interpolation polynomial of degree $\leq (n-1)$ at points $\left(s_j, g(s_j)\right)_{1\leq j\leq n}$

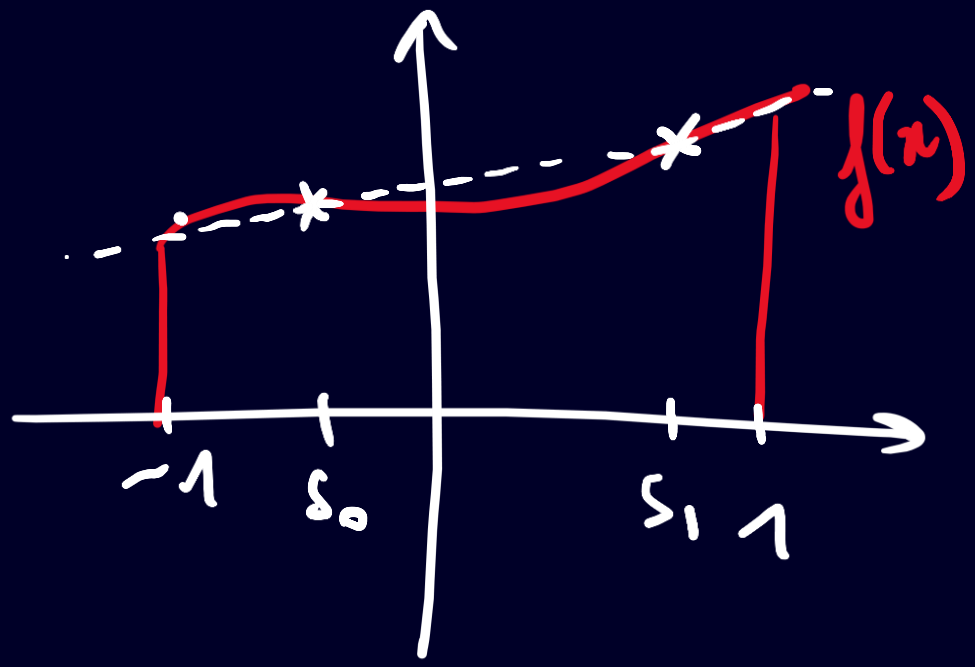
Lagrange

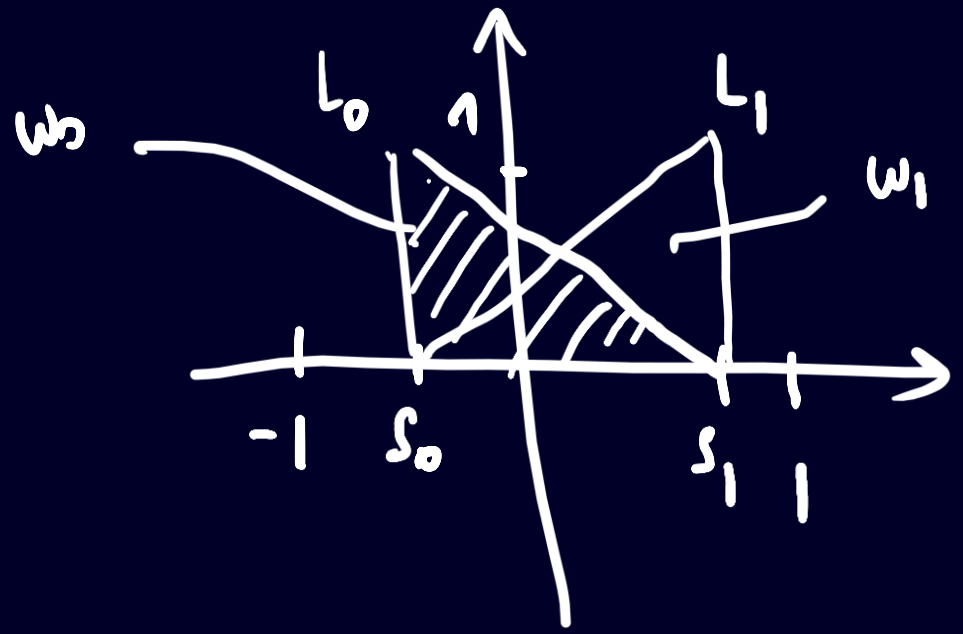
$$g(s) \simeq \sum_{j=1}^{n} g(s_j) L_i(s) \quad \text{with} \quad L_j(s) = \prod_{k=1, k\neq j}^{n} \frac{s - s_k}{s_j - s_k}$$

- Therefore, we have

$$\int_{-1}^{1} g(s)ds \simeq \int_{-1}^{1} \sum_{j=1}^{n} g(s_j) L_j(s) ds = \sum_{i=1}^{n} g(s_j) \int_{-1}^{1} L_j(s) ds = \sum_{j=1}^{n} w_j g(s_j)$$

Interpolating the integrand over [-1,1]

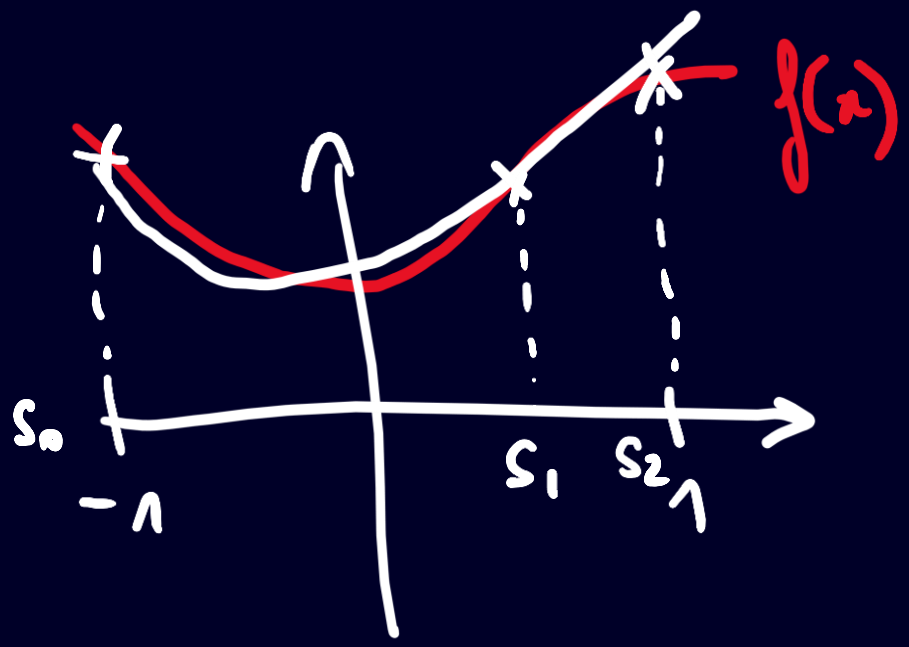


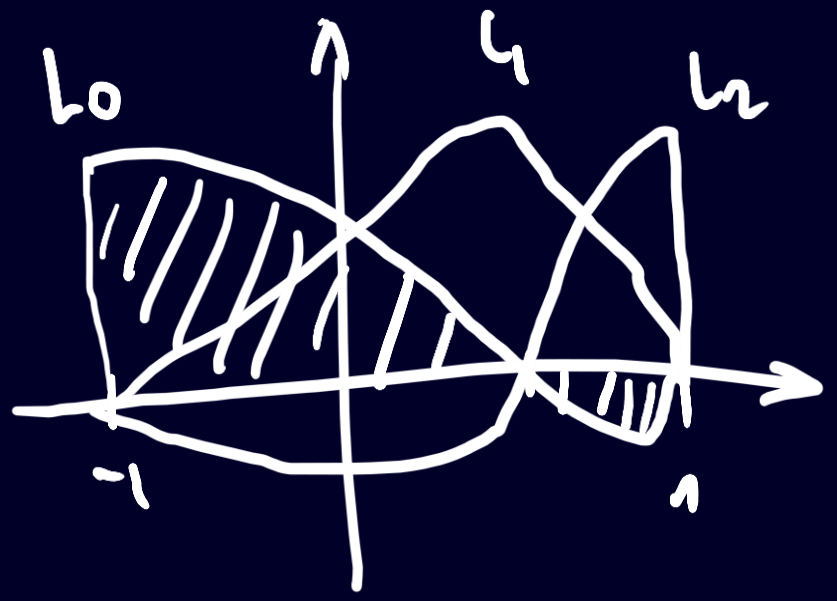


$$w_j = \int_{-1}^{1} L_j(s)\, ds$$

$$I = w_0\, g(s_0) + w_1\, g(s_1)$$

Interpolating the integrand over [-1,1]





$$w_j = \int_{-1}^{1} \alpha_j(s)\, ds \qquad I = w_0\, g(s_0) + w_1\, g(s_1) + w_2\, g(s_2)$$

Interpolating the integrand over [-1,1]

- We have obtained

$$\int_{-1}^{1} g(s)ds \simeq \sum_{j=1}^{n} w_j g(s_j) \quad \text{with} \quad w_j = \int_{-1}^{1} L_j(s)ds$$

- if g is a polynomial of degree $\leq n - 1$ then it coincides with its interpolation polynomial and the formula is exact.
- We have $\sum_{j=1}^{n} w_j = 2$ (because $\sum_{j=1}^{n} L_j(s) = 1 \rightarrow$ partition of unity)

Quadrature formula - definition

- The formula

$$\int_{-1}^{1} g(s)ds \simeq \sum_{j=1}^{n} w_j g(s_j) \quad \text{with} \quad w_j = \int_{-1}^{1} L_j(s)ds$$

- Is called **elementary quadrature formula** with n stages. The (s_j) are the **nodes** and the (w_j) are the **weights**
- By linearity the formula

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx = \sum_{i=0}^{N-1} \frac{h_i}{2} \int_{-1}^{1} f\left(\frac{h_i s + (x_i + x_{i+1})}{2}\right) ds$$

- Leads **composite quadrature formulas**

Quadrature formula

- **Remarks:**
  - Composite formulas can be obtained directly by replacing f by its *piecewise interpolant*
  - The interest of coming back to a fixed interval $[-1, 1]$ is to allow a unified treatment that is independent of the interval $[x_i\ , x_{i+1}]$
  - It shows that the weights (w_j) do not depend on i !

- We will now derive various quadrature rules thanks to interpolation

Example: Left-rectangle

- the **left-rectangle** formula (0th order polynomial with $x_0 = -1$)

$$\int_{-1}^{1} g(t)dt \simeq 2g(-1)$$

- Composite rule

$$\int_{a}^{b} f(x)dx \simeq \sum_{i=0}^{N-1} h_i f(x_i)$$

Example: Mid-point rule

- the **midpoint** formula (0th order polynomial with $x_0 = 0$)

$$\int_{-1}^{1} g(t)dt \simeq 2g(0)$$

- Composite rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N-1} h_i f\left(\frac{x_i + x_{i+1}}{2}\right)$$

Trapezoidal and Simpson's rules

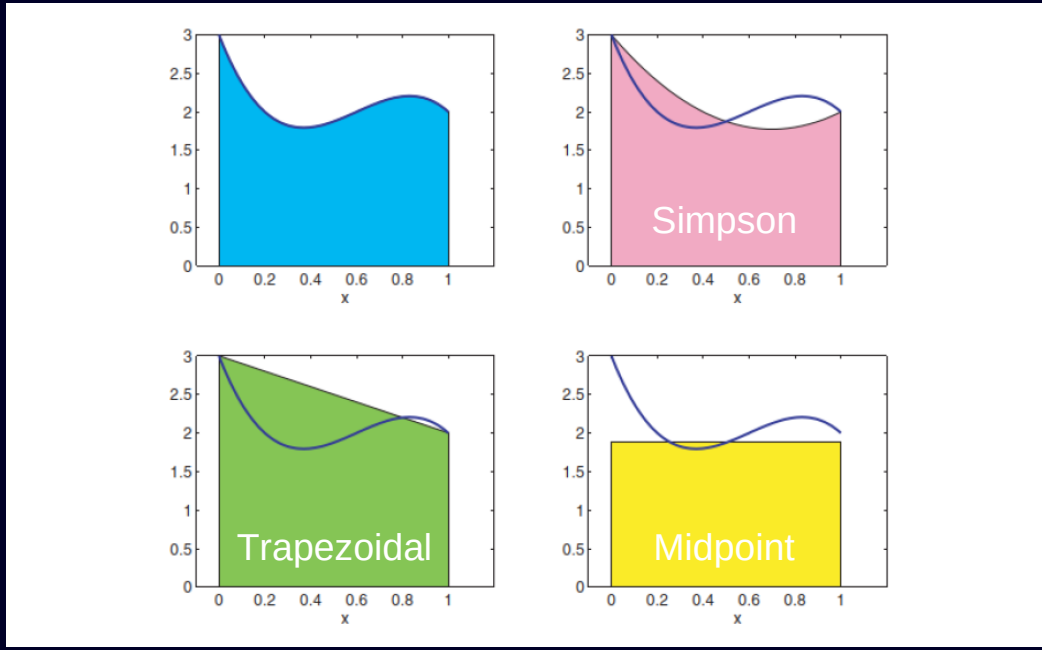
- the **trapezoidal** formula (1st order polynomial with $x_0 = -1, x_1 = 1$)

$$\int_{-1}^{1} g(t)dt \simeq 2\left(\frac{g(-1)+g(1)}{2}\right)$$

- the **Simpson's** quadrature rule (2nd order polynomial with $x_0 = -1, x_1 = 0, x_2 = 1$)

$$\int_{-1}^{1} g(t)dt \simeq 2\left(\frac{1}{6}g(-1)+\frac{4}{6}g(0)+\frac{1}{6}g(1)\right)$$

Elementary quadrature rules



Rule	Points	Weights
Left-rectangle	-1	2
Midpoint	0	2
Trapezoidal	-1, 1	1, 1
Simpson	-1, 0, 1	1/3, 4/3, 1/3

Exercise 1

Newton-Cotes formulae

- More generally, if we choose equidistant nodes (x_j) in the quadrature formulas $(n > 1)$

$$x_j = -1 + 2\frac{(j-1)}{(n-1)}, \quad 1 \leq j \leq n$$

- the quadrature formulae $\left(x_j, w_j\right)_{1 \leq j \leq n}$ are called **(closed) Newton-Cotes formulae**.

- for n $(n \geq 10)$ large, the weigths (w_j) explode and the signs are mixed which implies that the formulae are very sensitive to round-off errors
- some coefficients start to be such that $w_j < 0$ for $n \geq 9$
- This implies that the Newton-Cotes formulas are Unrestricted to $n \leq 8$

The Newton-Cotes formulas

n	w_j							name
2	$\frac{1}{2}$	$\frac{1}{2}$						trapezoidal
3	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$					Simpson
4	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$				Newton
5	$\frac{7}{90}$	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$			Boole
6	$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$		-
7	$\frac{41}{840}$	$\frac{216}{840}$	$\frac{27}{840}$	$\frac{272}{840}$	$\frac{27}{840}$	$\frac{216}{840}$	$\frac{41}{840}$	Weddle

Order of accuracy of elementary quadrature

- We say that a n stages elementary quadrature formula $\left(x_j\ , w_j\ \right)_{1\leq j\leq n}$ is of order p if the integration formula is exact for any polynomial of degree less or equal to $p\ -\ 1$
- By construction, a n stages elementary quadrature formula is at least of order n
- A n stages elementary integration formula $\left(x_j\ , w_j\ \right)_{1\leq j\leq n}$ is of order p if and only if

$$\forall\ 0 \leq q \leq p-1, \quad \sum_{j=1}^{n} w_j x_j^q = \begin{cases} \dfrac{2}{q+1} & \text{if } q \text{ even} \\ 0 & \text{if } q \text{ odd} \end{cases}$$

$$\int_{-1}^{1} x^q dx = \left[\frac{x^{q+1}}{q+1} \right]_{-1}^{1} = \frac{1^{q+1} - (-1)^{q+1}}{q+1}$$

An alternative approach

- An alternative method for deriving quadrature rules, is to choose the weights so that the rule integrates the first n polynomial basis functions exactly, which results in

$$
\begin{aligned}
w_1 \cdot 1 + w_2 \cdot 1 + \cdots + w_n \cdot 1 &= \int_a^b 1\,dx = b - a, \\
w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n &= \int_a^b x\,dx = (b^2 - a^2)/2, \\
&\vdots \\
w_1 \cdot x_1^{n-1} + w_2 \cdot x_2^{n-1} + \cdots + w_n \cdot x_n^{n-1} &= \int_a^b x^{n-1}\,dx = (b^n - a^n)/n.
\end{aligned}
$$

- This is the Method of Undetermined Coefficients

An alternative approach

- The matrix of this system is a **Vandermonde matrix**, which is nonsingular because the nodes x_i are assumed to be distinct

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b-a \\ (b^2-a^2)/2 \\ \vdots \\ (b^n-a^n)/n \end{bmatrix}$$

- The unique solution of this system yields the weights $w_1, \ldots, w_n$, which are the same as those given by integrating the Lagrange basis functions.

- Here you do not use Lagrange or ty to fit a polynomial, you just try to set the weights for fixed points

Exercise 2

Simpson's formula: checking our result

$$\int_{-1}^{1} g(t)dt \simeq 2\left(\frac{1}{6}g(-1)+\frac{4}{6}g(0)+\frac{1}{6}g(1)\right)$$

Simpson's quadrature

- Checking our results

- $q=0$

$$2\left(\frac{1}{6}\times 1+\frac{4}{6}\times 1+\frac{1}{6}\times 1\right)=2=\int_{-1}^{1}t^0dt$$

- $q=1$

$$2\left(\frac{1}{6}\times(-1)^1+\frac{4}{6}\times(0)^1+\frac{1}{6}\times(1)^1\right)=0=\int_{-1}^{1}t^1dt$$

- $q=2$

$$2\left(\frac{1}{6}\times(-1)^2+\frac{4}{6}\times(0)^2+\frac{1}{6}\times(1)^2\right)=\frac{2}{3}=\int_{-1}^{1}t^2dt$$

- So we see that the 3-stage Simpson rule is at least of order 3 (it integrates $p=2$)
- But what is interesting is that if we continue

- $q=3$

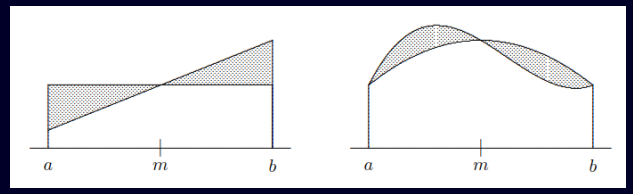
$$2\left(\frac{1}{6}\times(-1)^3+\frac{4}{6}\times(0)^3+\frac{1}{6}\times(1)^3\right)=0=\int_{-1}^{1}t^3dt$$

- $q=4$

$$2\left(\frac{1}{6}\times(-1)^4+\frac{4}{6}\times(0)^4+\frac{1}{6}\times(1)^4\right)=\frac{2}{3}\neq\frac{2}{5}=\int_{-1}^{1}t^4dt$$

- we see that the 3-stage Simpson rule is of **order 4** (it integrates $p=3$)
- This is a general property for symmetrical elementary quadrature formulas.

Symmetrical quadrature is always of even order



- An elementary quadrature formula $\left(w_j\,, x_j\right)_{1\leq j\leq n}$ is called symmetrical if and only if

$$x_j = -x_{n+1-j} \quad w_j = w_{n+1-j} \quad \forall\ 1 \leq j \leq n$$

- A symmetrical quadrature formula has always an **even order** (even with odd number of stages)
- If the formula is exact for polynomials of degree $\leq\ 2m\ -\ 1$ then it is automatically also exact for polynomials of order $\leq\ 2m$
- **In practice:**
  - The midpoint ($n\ =\ 1$) rule is of second order
  - The trapezoidal ($n\ =\ 2$) rule is also of second order
  - The Simpson's ($n\ =\ 3$) is of fourth order
  - The Newton-Cotes are symmetrical, so choosing n odd increases the order by one "for free"
  - This is why in quadrature tables, often only odd stages are shown

Exercise 3

Gaussian quadrature

- So far, the n nodes were prespecified and the n weights were chosen
- With only n parameters free to be chosen, the resulting degree is generally $n - 1$
- What if we now also optimize the location of the nodes?
  - That would lead to $2n$ free parameters $\left(x_j , w_j \right)_{1 \le j \le n}$
  - Maybe a degree of $2n - 1$ is achievable?

- In **Gaussian quadrature**, both the nodes and the weights are optimally chosen to maximize the degree of the resulting quadrature rule.
  - In general, for each n there is a unique n-point Gaussian rule, and it is of degree $2n - 1$.
  - Gaussian quadrature rules therefore have the **highest possible accuracy** for the number of nodes used, but they are significantly more difficult to derive than Newton-Cotes rules.

Gaussian Quadrature Rule

- To illustrate the derivation of a Gaussian quadrature rule, we will derive a two-point rule on the interval $[-1, 1]$, choosing w_1, x_1, w_2, x_2 up to order $2n - 1 = 3$

$$w_1 + w_2 = \int_{-1}^{1} 1 \, dx = 2$$

$$w_1 x_1 + w_2 x_2 = \int_{-1}^{1} x \, dx = 0$$

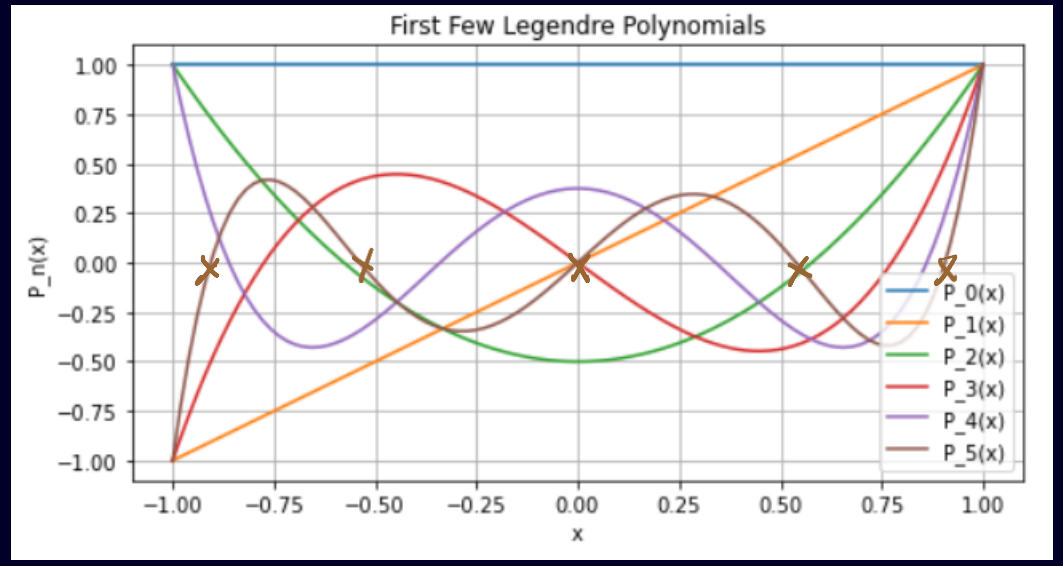
$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2 \, dx = \frac{2}{3}$$

$$w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^{1} x^3 \, dx = 0$$

One solution is given by $x_1 = -1/\sqrt{3}$, $x_2 = 1/\sqrt{3}$, $w_1 = 1$, $w_2 = 1$

Orthogonal polynomials

- Nodes of Gaussian quadrature rules are obtained from the zeroes of **orthogonal polynomials**.
- Gauss-Legendre polynomials, obtained from the roots of the Legendre polynomials are the most famous
- They are present in most scientific codes requiring integration



Orthogonal polynomials

- An orthogonal polynomials of order n is orthogonal to all polynomials of degree less than n (can you see why?)

$$\int_{-1}^{1} P_n(x) \cdot r(x) dx = 0$$

- Where $r(x)$ is any polynomial of degree $n - 1$
- This property is crucial because it allows us to construct optimal quadrature rules with $2n - 1$ degree

Why It Works for Degree $2n-1$

- Consider any polynomial $f(x)$ of degree $2n-1$. We can express $f(x)$ as

$$f(x) = q(x) + P_n(x) \cdot r(x)$$

- where $q(x)$ is a polynomial of degree less than n, and $r(x)$ is a polynomial of degree $n-1$
- When we integrate $f(x)$, we get

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} q(x)dx + \int_{-1}^{1} P_n(x) \cdot r(x)dx$$

$= 0$

- So

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} q(x)dx$$

- Since $q(x)$ is a polynomial of degree less than n, the Gaussian quadrature rule with n nodes corresponding to the roots of $P_n(x)$ exactly integrates $q(x)$, ensuring that $f(x)$ is also exactly integrated.

Gauss quadrature

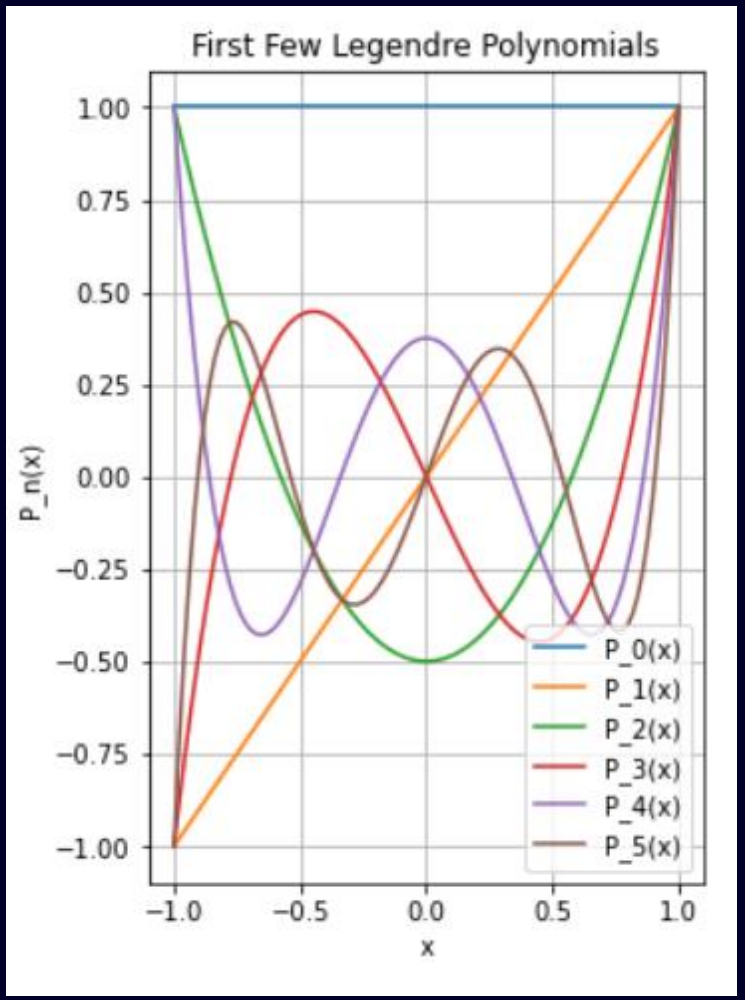
- **Legendre Polynomial Roots**:
  - Using n roots of the Legendre polynomial ensures we can integrate any polynomial up to degree 2n−1 *exactly*.
  - This minimizes error, making Gauss(-Legendre) quadrature the most efficient for polynomial integrals.

Gauss quadrature on K_a, order $2n-1=3$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.57735 02691 89625 76450 91488	1.00000 00000 00000 00000 00000

Gauss quadrature on K_a, order $2n-1=5$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.88888 88888 88888 88888 88889
2.	0.77459 66692 41483 37703 58531	0.55555 55555 55555 55555 55556



Exercise 4

Progressive Quadrature rules (Kronrod rules)

- **Purpose**:
  - Extend an existing Gauss quadrature rule by adding extra points to increase accuracy without changing the interval.
- **Concept**:
  - Start with an n −point Gauss quadrature rule.
  - Add $n + 1$ additional points to create a new rule with $2n + 1$ nodes (the Kronrod rule).
  - Kronrod rule is designed to keep all Gauss nodes and add new ones to improve accuracy
- **Applications**:
  - Efficient error estimation: Combine the Gauss and Kronrod rules to approximate the integral and estimate the error from their difference.
  - Commonly used in adaptive quadrature algorithms to control accuracy.
- **Example**:
  - The Gauss-Kronrod (G_7, K_{15}) rule uses 7 Gauss points and extends to 15 points for a higher accuracy approximation and error estimation.

Alternative Gaussian quadratures

- Gaussian quadrature do not include **end points,** in some occasions it is useful

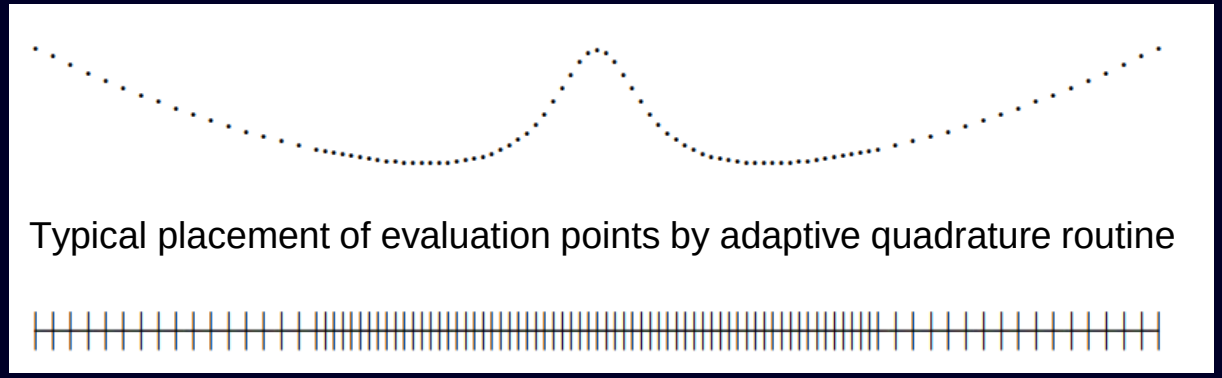
- **Gauss-Radau:**
  - In Gauss Radau quadrature, one of the endpoints of the interval of integration is prespecified as a node, leaving the remaining $n - 1$ nodes, as well as all n weights, free to be chosen to maximize the degree of the resulting rule
  - There are $2n - 1$ free parameters, so an n-point Gauss-Radau rule has degree $2n - 2$

- **Gauss-Lobatto:**
  - In Gauss-Lobatto quadrature, both endpoints of the interval of integration are prespecified as nodes, leaving the remaining $n - 2$ nodes, as well as all n weights, free
  - There are $2n - 2$ free parameters, so an n-point Gauss-Lobatto rule has degree $2n - 3$

Adaptive quadrature rule

- **Adaptive Quadrature** is a composite rule which refines the interval of integration selectively, focusing more sampling points in areas where the integrand is difficult to approximate.

1. **Error Estimation**: Use two quadrature rules, Q_{n_1} and Q_{n_2}, whose difference estimates the error.
   - **Trapezoid and Midpoint Rules**: Basic
   - **Higher-Degree Rules**: More efficient, e.g., Gauss-Kronrod (G7, K15)
   - **Simpson's Rule**: Uses a single rule at two levels of subdivision.
2. **Progressive Rule Application**:
   - Apply both rules on the initial interval $[a, b]$
   - If the estimated error exceeds tolerance, divide the interval and repeat on each subinterval.
   - Continue until all subintervals meet the desired accuracy tolerance.



Typical placement of evaluation points by adaptive quadrature routine

Improper Integrals

- If either the integrand or the interval is unbounded, then it may still be possible to define an improper integral
- For an integrand with a singularity (i.e. unbounded at a point c ∈ [a, b]), the improper integral is defined as

$$\int_a^b f(x)\,dx = \lim_{\gamma \to c-} \int_a^\gamma f(x)\,dx \; + \; \lim_{\gamma \to c+} \int_\gamma^b f(x)\,dx,$$

- Provided that f is integrable on any subinterval and the indicated limits exist and are finite.
- Best approach for dealing with a singularity in the integrand is to remove the singularity either by
  1. transforming the variable of integration
  2. dividing out / subtracting off an analytically integrable function having the same singularity

Exercise 5

Double Integrals

- There are different strategies

$$\int_a^b \int_c^d f(x,y)\,dx\,dy.$$

  - Use a pair of adaptive one-dimensional quadrature routines, one for the outer integral and the other for the inner integral. Each time the outer routine calls its integrand function, the latter in turn calls the inner quadrature routine.

  - Use a Cartesian product rule. Such rules result from applying one-dimensional quadrature rules in successive dimensions. This approach is limited to regions that can be decomposed into rectangles.

  - Use a nonproduct interpolatory cubature rule. Such rules, with error estimates, are available for a number of standard regions, the most important of which for adaptive use is triangles, since many two-dimensional regions can be efficiently triangulated to any desired level of refinement.

Multiple Integrals

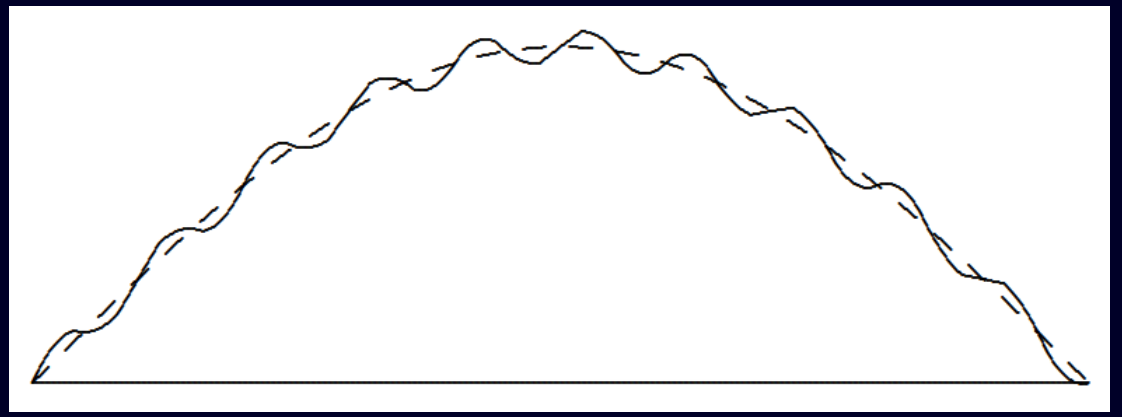
- **Purpose**: Efficiently compute integrals in high-dimensional spaces
- **Concept**:
  - Sample the function at n random points within the integration domain.
  - Get the average of these function values & multiply by volume of the domain to estimate the integral.
- **Advantages**:
  - **Scales well with dimension**: Unlike traditional methods, Monte Carlo integration's cost does not grow exponentially with the number of dimensions.
  - **error reduction**: The error decreases as $\frac{1}{\sqrt{n}}$, where n is the number of sample points.
- **Application**:
  - Widely used in fields requiring high-dimensional integration, such as statistical physics, finance, and machine learning.

Exercise 6

2. Numerical Differentiation

Numerical differentiation

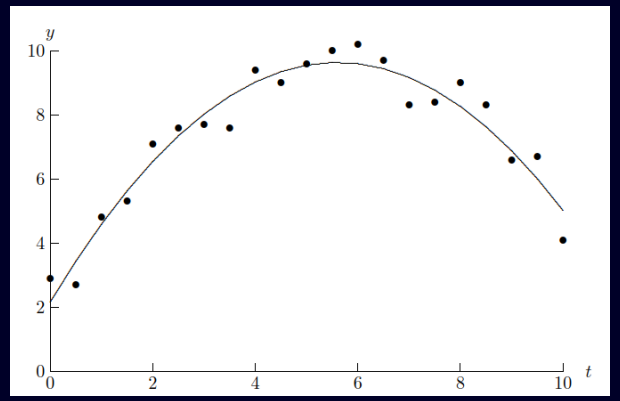
- Differentiation is an inherently sensitive problem: small data perturbations → large changes in the result
- Integration, on the other hand, is a smoothing process and is inherently stable in this respect.
- This contrast is not surprising, since they are inverse processes to each other
- Consider the function below and a perturbed/noisy version of it



- ➔ Integral estimate will only be marginally affected, whereas derivatives estimates will be drastically modified

Interpolation vs smoothing

- Main approach for derivative approximation consists in fitting a smooth function to discrete data points, then differentiate this function to estimate derivatives
- **Interpolation vs smoothing:** one should use interpolation if data is smooth; for noisy data, a smoothing approach (like least squares polynomials or splines) is preferable.



- Here, would you interpolate or would you smooth?
  - If the 21 data points were fit exactly by a polynomial of degree 20, derivatives would be erratic…
  - By contrast derivative of the quadratic polynomial is well behaved and insensitive to data changes

Finite Difference

- Although not appropriate for noisy data, finite differences are good for smooth functions known analytically or defined implicitly by a **differential equation**

- Given a smooth function $f: \mathbb{R} \to \mathbb{R}$, we wish to approximate its first and second derivatives at a point x.

  - For a given step size h, consider the Taylor series expansions

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \cdots$$

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \cdots.$$

Finite difference

- We *obtain* the *forward difference formula* and the *backward difference formula* (first-order accurate $O(h)$)

(1)

$$\begin{aligned} f'(x) &= \frac{f(x+h)-f(x)}{h} - \frac{f''(x)}{2}h + \cdots \\ &\approx \frac{f(x+h)-f(x)}{h}, \end{aligned}$$

(2)

$$\begin{aligned} f'(x) &= \frac{f(x)-f(x-h)}{h} + \frac{f''(x)}{2}h + \cdots \\ &\approx \frac{f(x)-f(x-h)}{h}, \end{aligned}$$

- Combining them yields the *centered formula* (second-order accurate $O(h^2)$)

(1) + (2)

$$\begin{aligned} f'(x) &= \frac{f(x+h)-f(x-h)}{2h} - \frac{f'''(x)}{6}h^2 + \cdots \\ &\approx \frac{f(x+h)-f(x-h)}{2h}, \end{aligned}$$

- *Centered formula* for the second derivative (second-order accurate $O(h^2)$)

(1) − (2)

$$\begin{aligned} f''(x) &= \frac{f(x+h)-2f(x)+f(x-h)}{h^2} - \frac{f^{(4)}(x)}{12}h^2 + \cdots \\ &\approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2}, \end{aligned}$$

Exercise 7

Automatic Differentiation

- For any function expressed by a computer program, alternative is automatic differentiation
- A computer program = basic arithmetic operations **&** simple functions
- Thus, derivatives can be propagated through the program by repeated use of **the chain rule** step by step along with the function itself
- The result is the true derivative of the original function, subject only to rounding error but suffering no discretization error.
- Several effective software packages are now available for automatic differentiation
  - Some of these packages accept a Fortran or C input program and then output a second program for computing the desired derivatives
  - Other packages use operator overloading to perform derivative computations automatically along with the function evaluation.
- Allows determining the sensitivity of the output of a program to perturbations in its input parameters

3. Summary

Summary

- **Numerical integration:**
  - Elementary **quadrature rules** typically yield $n-1$ order accuracy
  - Gaussian rules yield $2n-1$ order by optimizing both quadrature weights and points
  - Dividing original domain (composite rules) possibly with adaptivity is often the best choice
  - For high-dimensional domains, Monte Carlo integration is the method of choice
- **Numerical differentiation**:
  - Highly sensitive to noise in the data (by contrast to integration)
  - For smooth known or implicit functions (PDE), **finite difference** formulas are typically used
  - Obtained from Taylor expansion (forward, backward, centered) or from polynomial interpolation